Gestión de consumos de clientes en plataforma SaaS Wattwin

Adrián Módenes Zapico

Resumen– La empresa de servicios tecnológicos SIIE hace ya tiempo que planeaba automatizar el proceso de gestión del consumo y suscripciones de sus clientes de la plataforma SaaS (Software como servicio) Wattwin, ya que siempre ha sido un proceso que se realizaba de forma manual. Por eso mismo, de la mano de su equipo de desarrollo y utilizando los conocimientos de programación web adquiridos en el grado, este proyecto ha abordado la creación de un sistema automatizado de gestión del consumo. Después de su implementación, se ha logrado proporcionar al cliente un control autónomo sobre su consumo, facilitando la gestión interna a una empresa aún en fases tempranas de crecimiento. Se espera que en breve toda esta nueva automatización llegue a producción, centrando el trabajo en otros proyectos vinculados a la gestión del consumo, pudiendo así abarcar nuevos casos de uso y explorar otras posibilidades de diseño.

Palabras clave– SIIE, Wattwin, FrontEnd, BackEnd, Suscripciones, Chargebee, Espacio de trabajo, Dominio

Abstract– The technology services company SIIE had been planning for some time to automate the process of managing the consumption and subscriptions of its customers of the SaaS (Software as a Service) Wattwin platform, as it has always been a manual process. For this reason, with the help of its development team and using the knowledge of web programming acquired in the degree, this project has tackled the creation of an automated consumption management system. After its implementation, it has managed to provide the client with autonomous control over its consumption, facilitating the internal management of a company still in the early stages of growth. It is expected that all this new automation will soon be in production, focusing the work on other projects linked to consumption management, thus being able to cover new use cases and explore other design possibilities.

Keywords– SIIE, Wattwin, FrontEnd, BackEnd, Subscriptions, Chargebee, WorkSpace, Domain

1 Introducción

Las empresas energéticas llevan ya muchos años buscando alternativas a las energías tradicionales, con el fin de poder ofrecer a sus clientes servicios a medida basados en energías renovables. De esa necesidad apareció SIIE [1], una firma independiente de consultoría y servicios tecnológicos con sede en Barcelona y compuesta por un equipo altamente cualificado con más de 20 años de trayectoria, tanto en energías renovables como en negocios BPO/BPM, es decir, basados en subcontratación de funciones de procesos de negocio a proveedores de servicio. SIIE es una empresa joven e innovadora en el mercado, la cual ofrece servicios de desarrollo empresarial y soporte técnico en el diseño, construcción, financiamiento y operación de activos de generación de energía basados en tecnologías renovables, en concreto energía solar fotovoltaica.

Uno de los problemas principales de las instalaciones fotovoltaicas es la alta complejidad de tanto el diseño como la planificación y su posterior instalación. Es por eso, que SIIE se centró en buscar una manera de centralizar todos los procesos involucrados en su adquisición, facilitando así el trabajo de los diseñadores y proporcionando una interfaz amigable y personalizable para sus clientes.

Para poder ofrecer estos servicios de manera óptima, se decidió desarrollar la plataforma Wattwin [2]. Esta plataforma es un Software como Servicio (SaaS), accesible a través de Internet, creado para el diseño y planificación de instalaciones fotovoltaicas, el cual actúa de intermediario entre la empresa proveedora de servicios energéticos y los clien-

• E-mail de contacto: adrian.modenes@e-campus.uab.cat
• Mención realizada: Tecnologies de la Informació
• Trabajo tutorizado por: Ramon Martí Escalé (TI)
• Curso 2020/21

tes. Los servicios que ofrece se centran tanto en desarrollar proyectos totalmente nuevos a escala de servicios públicos, hasta la gestión de activos de plantas de energía operativas. Consiguiendo así proporcionar una descripción comercial global e independiente a inversores públicos y/o privados que aspiran a convertirse en actores del sector energético.

Wattwin es una plataforma aún en fases tempranas de desarrollo, la cual carecía de automatización de procesos básicos como era el consumo de los clientes. Por este motivo, se decidió desarrollar el Módulo de Gestión del Consumo (MGC), el cual automatiza todo el proceso de gestión del consumo así como la facturación del mismo. La realización de este proyecto ha ayudado mucho en la evolución del servicio web, dando además la posibilidad de guardar un registro y un control sobre las acciones de los clientes dentro de la plataforma.

En lo que resta de documento, se explicarán los objetivos iniciales del desarrollo del MGC de la plataforma Wattwin, el estado del arte, la metodología y planificación que se ha seguido en su desarrollo, se explicarán cada una de las tareas desarrolladas, se expondrán y discutirán los resultados y, por último, las conclusiones cerrarán el trabajo.

2 OBJETIVOS

La finalidad de desarrollar el MGC ha sido la de conseguir una nueva e importante funcionalidad en la plataforma que permita automatizar el proceso de facturación y poder tener una jerarquía de los diferentes clientes y ofrecerles diferentes servicios dependiendo del nivel y plan que posean, así como ofrecerles, según el plan contratado, un seguimiento de su consumo por cada WorkSpace (espacio de trabajo), ya que cada suscripción de usuario va asociada a un espacio de trabajo diferente. También, se le proporciona al cliente información de interés relacionada con su consumo, para así facilitar las gestiones y visualización de sus datos, creando paralelamente un sistema de facturación y visualización del consumo total del cliente, con su respectiva vinculación bancaria para los cobros de las tarifas.

En el siguiente listado se exponen y explican los diferentes objetivos escogidos para este proyecto, ordenados por prioridad, con sus respectivos subobjetivos.

- **Definición proyecto**. Se definen los requisitos, objetivos y casos de uso del proyecto, así como la creación del diagrama de Gantt en base a la planificación seguida, y los Mockups para dar soporte en la definición de la vista.
  - **Identificación de requisitos**. Primero de todo, es necesario definir los requisitos del proyecto, tanto funcionales como no funcionales, para tener una idea clara y concisa sobre la estructura y funcionalidades necesarias a desarrollar.
  - **Definición objetivos**. Se definen los objetivos del proyecto en base a los requisitos extraídos en la fase anterior. Estos objetivos dan una visión global de los resultados esperados tras el desarrollo del proyecto.
  - **Definición casos de uso**. En función de cada requisito funcional y no funcional se define una jerarquía de funcionalidades necesarias para el correcto funcionamiento del MGC.
  - **Creación diagrama de Gantt**. Se crea un diagrama con la planificación para el proyecto con los periodos de cada una de las tareas.
  - **Creación Mockups**. Se crean los Mockups necesarios para tener una visión más definida de la interfaz a ofrecer en el MGC.
- **Definición de planes**. Se define la jerarquía de planes de suscripción junto con las limitaciones de cada uno y su tarifa vinculada.
  - **Definición de jerarquía de planes de suscripción**. Se crea una jerarquía de los distintos planes contratables, junto con las funciones internas de los mismos.
  - **Definición de acciones por plan**. Se definen las funcionalidades restringidas y exclusivas entre los diferentes planes, haciendo la distinción entre las acciones de los administradores de WorkSpace y los administradores de Wattwin, los cuales pueden realizar acciones adicionales a los primeros.
  - **Generación de tarifas vinculadas a cada plan**. Se definen las tarifas temporales de cada plan, ya que cada uno debe tener una tarifa fija con un coste por ítem de consumo vinculado, si se llega a consumir más del fijado en el plan se añadirá como extra al coste final, variando el coste en función del plan contratado.
- **Gestión de suscripción**. Se crea un entorno que permite al cliente suscribirse, modificar la suscripción o cancelarla, así como limitar las operativas disponibles en función del tipo de suscripción, e incluso de los permisos que posea. Su desarrollo se ha basado en crear los modelos de datos que permiten poder guardar íntegramente los datos de suscripción, crear los servicios necesarios del BackEnd, personalizar la interfaz de gestión de la suscripción y los componentes para la vista, y finalmente, testearlo.
- **Integración Chargebee**. Se integra Wattwin con el software de facturación Chargebee [3]. Este software permite gestionar las suscripciones, los planes y su vinculación con los clientes y la facturación, entre otros. Este proceso ha comportado el estudio de su funcionamiento y la forma de integrarlo con Wattwin, la creación de *endpoints* (URLs del BackEnd que responden a peticiones), la creación de un registro de actividad sobre las acciones realizadas en Chargebee, y para acabar, el testeo de esta nueva parte.
- **Gestión de consumo**. Se desarrolla el servicio que permite al cliente la gestión de su consumo, por lo que él mismo podrá consultar el consumo actual y el historial de consumos antiguos, vinculados con el coste. Este proceso ha comportado la configuración del control de facturación, la definición de los tipos de consumo e implementación del control de consumo, y por último, el testeo.

- **Testing**. Se integra todo el código de las diferentes funcionalidades y servicios desarrollados para garantizar su correcto funcionamiento en conjunto. Para ello, se han definido y ejecutado pruebas de testeo, con la finalidad de encontrar errores y solucionarlos.
  - **Definición de pruebas de testeo**. Se definen las pruebas a realizar contra la propia funcionalidad desarrollada, con la finalidad de encontrar errores de navegación o bugs internos.
  - **Ejecución de pruebas de testeo**. Una vez definidas las pruebas de testeo a realizar, se ejecutan para conseguir encontrar fallos en el servicio.
  - **Solución bugs**. Se analizan y corrigen todos los problemas encontrados en la *Ejecución de pruebas de testeo*, generando así una versión del código final sin errores ni *bugs*.

  Se debe tener en cuenta que el testeo realizado en las tareas anteriores ha sido para garantizar que el resultado obtenido en cada una de ellas es el previsto, pero en el caso de esta tarea, ha sido enfocada en el buen funcionamiento de las diferentes funcionalidades en conjunto.

- **Creación paper final y presentación**. Una vez finalizado el proyecto, se ha realizado y entregado la versión final del paper con una explicación global de principio a fin del desarrollo del proyecto y los resultados obtenidos, así como una presentación explicativa del proyecto para ser expuesta delante de un tribunal, para su posterior aprobación.

3 Estado del arte

Actualmente, los servicios que ofrece Wattwin no están todos automatizados, por lo que es necesario empezar a automatizar las funcionalidades básicas como ha sido el caso de la gestión del consumo, ayudando a quitar trabajo manual. Esta automatización es la que puede llevar a la empresa a un mayor reconocimiento, ya que es una funcionalidad básica para un servicio fundamentalmente basado en la venta de un producto técnico. Este servicio automático es presente en la mayoría de competencias en el mercado, por lo que carecer de ella puede suponer un retroceso en la evolución del servicio web.

No obstante, no existen ninguna empresa nacional que trabaje como SIIE, si que existen empresas que están enfocadas en la energía solar fotovoltaica, pero no que ofrezcan servicios de Outsourcing (BPO) y Gestión de Procesos de Negocio (BPN) para soporte técnico y de gestión en la industria solar fotovoltaica, proporcionando servicios basados en web y aplicaciones para el diseño, la autorización y la monitorización de sistemas solares fotovoltaicos, ayudando así a proporcionar a sus clientes servicios eficientes y actualizados que actúan de intermediario entre la empresa proveedora de electricidad y el cliente.

Para desarrollar el MGC, se ha trabajado según el CBA (Component Based Architecture - Arquitectura basada en componentes) [4] para estructurar y ordenar el código del Frontend, la cual está actualmente en auge. Los lenguajes utilizados han sido TypeScript [5], que es un superconjunto de JavaScript que principalmente añade tipos estáticos y objetos basados en clases, que a su vez es utilizado en SIIE sobre el *framework* Angular 8 [6] de JavaScript. También Pug [7], que es un motor de plantillas que hace más rápida la codificación de HTML, con la ayuda de Sass [8], que es un lenguaje de script que se traduce a CSS, haciéndolo más fácil de codificar.

En el caso del código del Backend, se ha utilizado también TypeScript con Angular 8, pero añadiendo promises [9], que son útiles para trabajar con operaciones asíncronas, es decir, en donde las ejecuciones no son secuenciales. Además, se utiliza el framework LoopBack 3 [10], que trabaja sobre Nodejs y TypeScript para crear APIs y microservicios.

Todos estos lenguajes no han sido escogidos arbitrariamente, sino que son la evolución de sus antiguos o *frameworks* que facilitan su implementación. Como ejemplo claro, están las bases de datos utilizadas, las cuales son MongoDB [11] y el software de visualización de datos para Elasticsearch, Kibana [12], en donde las dos son bases de datos NoSQL, es decir, que trabajan de forma distribuida, permitiendo una mejor escalabilidad y velocidad de procesamiento.

Como software de desarrollo, se ha contado con el software de facturación de suscripciones y operaciones de ingresos conocido como Chargebee, al cual se le ha volcado el proyecto, ya que se han aprovechado muchas de las funcionalidades que ofrece para automatizar aún más la gestión de las suscripciones y los consumos vinculados.

4 Metodología

SIIE, al igual que muchas otras empresas de su sector, trabaja con metodología AGILE, la cual consiste en dividir cada proyecto en pequeñas tareas que tienen que completarse y entregarse en periodos cortos de tiempo. De esta manera, al dividirse el trabajo, es más fácil llevar un control sobre las tareas realizadas, así como una mejor organización, en caso de trabajar en equipo. Esta metodología también implica reuniones matinales de 15-30 minutos entre los desarrolladores para comentar el trabajo a realizar durante ese día.

Para registrar las tareas, se ha utilizado una herramienta gratuita en linea llamada Trello, la cual permite organizar las tareas como tarjetas por secciones, las cuales representan los diferentes *Sprints*, y en donde cada una de ellas puede ser rellenada con información sobre la tarea y puede ser asignada a una o más personas, creando así bloques cerrados de plazo fijo. Trabajar con ella ha permitido dinamizar el trabajo, ya que el hecho de tener una plataforma de tareas común para todos da facilidades a la hora de organizar los trabajos, tanto individuales como grupales.

Durante el desarrollo de este proyecto, todo el trabajo se ha realizado en línea, utilizando la aplicación Microsoft Teams para amenizar el teletrabajo. Gracias a él, se ha permitido la comunicación tanto por mensaje privado con cualquiera de los integrantes del equipo, como a través de chats y/o llamadas grupales.

Durante el desarrollo del MGC, se ha contado con la ayuda de un integrante del equipo de desarrollo que ha estado disponible por si surgía cualquier tipo de dudas o problemas. También, a la hora de subir el código al repositorio,

se ha tenido que crear una *pull-request* que ha sido revisada por un supervisor de código, el cual ha solicitado cambios en caso de errores o código pobre. Así como las distintas fases posteriores que conforman una subida a producción, en donde se testean las nuevas funcionalidades en distintos entornos de test. Por lo que se puede ver, existe un filtro humano suficiente como para garantizar que el código subido es correcto, aunque no garantiza al cien por cien que no existan errores aislados, ya que, si se encuentran, se debe volver a la fase inicial para arreglarlo, o realizar un *hotfix* (parche en caliente).

5 PLANIFICACIÓN

En este proyecto, como se ha explicado anteriormente, se ha trabajado con la metodología AGILE, en donde se han definido 7 tareas, correspondientes a los 7 objetivos iniciales, abordando periodos de varias semanas. Algunas de estas tareas se dividen en subtareas más detalladas, correspondientes a los subobjetivos, las cuales abordan periodos más reducidos, de entre una y dos semanas.

Gran parte de los periodos de las tareas y subtareas no han variado en gran medida respecto a la planificación inicial, excepto las que dependían de decisiones de diseño empresarial, como han sido las subtareas de *Definición de planes*, que se han visto alteradas a lo largo del desarrollo, y algunas variaciones en los casos de uso debido a ligeros cambios en los objetivos del proyecto.

En la Fig.1 se puede ver el diagrama de Gantt de las tareas de este proyecto junto con sus periodos.

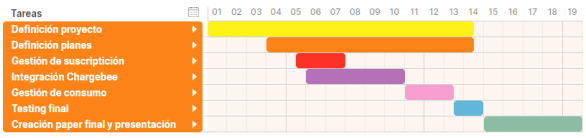

Fig. 1: *Diagrama de Gantt de las tareas.*

6 DESARROLLO

En esta sección se explica la fase de desarrollo del proyecto, en la cual se explican todas las tareas que lo conforman, las cuales son *Definición proyecto*, *Definición planes*, *Gestión de suscripción*, *Integración Chargebee*, *Gestión de consumo* y *Testing final*, exceptuando la tarea de *Creación paper final y la presentación*.

6.1 Definición proyecto

En este apartado se explica el trabajo realizado durante el desarrollo de las subtareas que conforman la definición del proyecto, las cuales son *Identificación de requisitos*, *Definición casos de uso*, *Creación diagrama de Gantt* y *Creación Mockups*, exceptuando la Definición de objetivos, que ya ha sido explicada en la sección 2 de este documento.

6.1.1 Identificación de requisitos

Como primera tarea del proyecto, su desarrollo se ha llevado a cabo en reuniones con el equipo de trabajo, que han sido enfocadas en definir bien los requisitos tanto funcionales como no funcionales.

- **Requisitos funcionales**
  - Definir jerarquía de suscripciones.
  - Definir los distintos planes.
  - Definir las acciones disponibles por plan.
  - Definir las tarifas vinculadas a cada plan.
  - Permitir al cliente suscribirse.
  - Permitir realizar cambios en la suscripción.
  - Permitir visualizar los datos de suscripción al cliente.
  - Registrar las acciones realizadas por un cliente suscrito.
  - Mantener la base de datos actualizada con los datos de consumo de los clientes.
  - Permitir visualizar y descargar las facturas.
  - Integrar Wattwin con una plataforma de gestión de cobros.
  - Permitir visualizar al cliente información de su consumo.
- **Requisitos no funcionales**
  - Desarrollar el servicio con los lenguajes y metodologías estándares de la empresa.
  - Mantener la lógica de la plataforma en las interfaces y funcionalidades desarrolladas.
  - Proporcionar mensajes de error intuitivos para cuando una acción no se pueda llevar a cabo.

6.1.2 Definición casos de uso

En este tarea se han definido los diferentes diagramas de casos de uso que abarcan todos los objetivos del proyecto. Se encuentran divididos entre las acciones que puede realizar un administrador de Wattwin, que es la persona con acceso interno a todas las suscripciones, y el administrador del WorkSpace, que es quien ha contratado los servicios.

En la sección A.1 del apéndice se pueden observar estos diagramas junto con una pequeña explicación de los mismos.

6.1.3 Creación diagrama de Gantt

Llegado el momento de empezar a planificar las tareas del proyecto, se ha desarrollado un diagrama de Gantt, en donde se han intentado cubrir todos los objetivos ya definidos en tareas con periodos razonables. Este diagrama se ha ido adaptando a nuevos requisitos y cambios del diseño empresarial que han ido surgiendo a lo largo del desarrollo. Igualmente, el diagrama ha mantenido su estructura general y gran parte de sus periodos.

La diferenciación entre el diagrama inicial y el final puede ser observada en la sección A.2 del apéndice.

6.1.4 Creación Mockups

Esta tarea, se ha visto afectada por los cambios de requisitos del desarrollo, es decir, por los cambios de diseño y de interfaces que ha comportado la integración del software de facturación Chargebee, que es explicado con más detalle más adelante. Por lo que algunas de las interfaces que inicialmente se creían necesarias, han sido sustituidas por las que proporciona este software.

Igualmente, se ha requerido de la creación de dos Mockups correspondientes a la visualización de la *card* de *Mi suscripción* y al *pop up* de suscripción, que también serán explicados más adelante. El primero Fig.2, muestra la página del detalle de un Dominio, en donde se puede ver un menú laterial a la izquierda con las diferentes secciones de la plataforma. En la de *Mi Dominio* (la cual actualmente corresponde a *Mi WorkSpace*), se pueden visualizar en la primera *card* los datos del Dominio, y en la segunda los datos privados de suscripción, además de acceder al portal de facturación de Chargebee. Las dos "tarjetas" rectangulares representan lo que en este documento se cataloga como *card*, y es donde siempre se van a mostrar los datos. El segundo Fig.3, muestra el *pop up* o "ventana" que se abre al clicar sobre el botón de suscribirse de la *card* de *Mi suscripción*, el cual es únicamente visible si el Dominio no tiene una suscripción, y sirve para seleccionar un plan de suscripción y en consecuencia suscribir al Dominio.

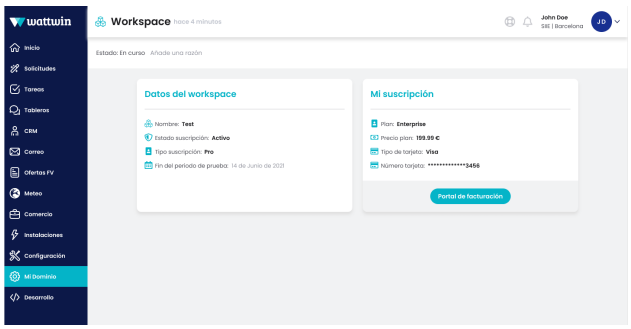

Fig. 2: *Mockup detalle de un Dominio.*

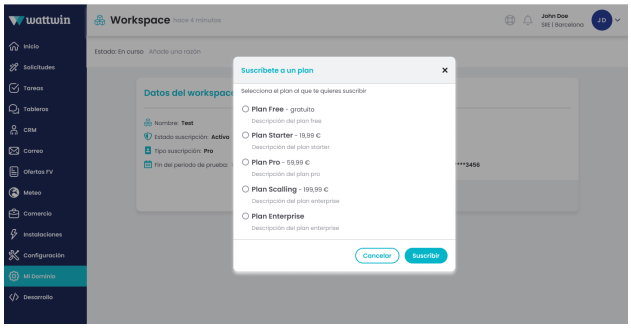

Fig. 3: *Mockup pop up suscripción.*

6.2 Definición planes

En este apartado se explica el trabajo realizado durante el desarrollo de cada una de las subtareas que han conllevado la definición de los planes, es decir, de los *items* a los que un usuario se puede suscribir a la plataforma. Estas tareas son *Definición acciones por plan*, *Generación tarifas vinculadas a cada plan* y *Generación tarifas vinculadas a cada plan*.

6.2.1 Definición jerarquía de planes de suscripción

Una vez se ha empezado a profundizar en como sería implementada la gestión de las suscripciones, todo el equipo de trabajo decidió crear los siguientes planes, los cuales están ordenados de más económico a más caro:

- **Free**. Plan gratuito, con acciones muy limitadas.
- **Starter**. Plan base, el cual representa el más económico y está enfocado para usuarios nuevos.
- **Professional**. Plan Profesional, más avanzado que el *Starter* y enfocado a usuarios que tienen claro el producto que se ofrece.
- **Scaling sales**. Plan de venta escalar, nivel más alto de plan contratable para usuario medio.
- **Enterprise**. Plan enfocado a empresas o socios.

6.2.2 Definición acciones por plan

Una vez definidos los planes, se han vinculado con las acciones que tienen disponibles. Finalmente, las restricciones no han sido aplicadas, ya que se desea tener una gestión global del consumo funcional en la plataforma, y partiendo de ahí, empezar a aplicarlas. Igualmente, se han definido dos acciones restringibles, una de las cuales solo estará disponible para los planes por encima del *Professional*, incluyéndolo, la cual corresponde a el *Recomendador de potencia instalada*, y la otra solo para el *Scaling sales* y el *Enterprise*, la cual corresponde al *Análisis de sensibilidad*.

6.2.3 Generación tarifas vinculadas a cada plan

Para la vinculación de tarifas con los planes, se debe tener en cuenta que la jerarquía de precios sigue el orden del listado del apartado 6.2.1, por lo que los precios tienden a más altos contra más abajo estén del listado, pudiendo variar con el tiempo, ya que dependen de decisiones empresariales.

Los precios exactos no pueden ser proporcionados, ya que, de momento, son privados para empresa, a la espera de tener el servicio disponible para los usuarios finales.

6.3 Gestión de suscripción

En este apartado se explica el trabajo realizado durante el desarrollo de cada una de las subtareas de la gestión de la suscripción, las cuales son *Creación modelo de datos*, *Identificación y creación del servicio*, *Diseño de interfaz y componentes* y *Testing*.

6.3.1 Creación modelo de datos

Primero de todo, para poder guardar en base de datos información de las suscripciones, se ha creado el modelo de datos *DomainSubscription* con los campos siguientes:

- **customerId**. Campo obligatorio que corresponde a la id del cliente, la cual es proporcionada por Chargebee en el momento de crear una suscripción.

- **accessLevel**. Campo obligatorio que corresponde al nivel de acceso al servicio. El rango del *accessLevel* ha sido definido entre 0 y 500, y será el encargado de restringir las operativas en función del plan contratado, los cuales tendrán asignado un *accessLevel* cada uno.

- **subscriptionType**. Campo obligatorio que corresponde al tipo de suscripción, el cual contiene el nombre del plan contratado.

- **trialExpirationDate**. Campo opcional que corresponde a la fecha en la que caduca el período de prueba.

- **addOn**. Campo opcional (añadido durante el desarrollo de la tarea de *Gestión de consumo*) que contiene los *addOns* variables, en concreto su id, tipo, uso y fecha de último uso. Cada vez que se realiza un cambio entre planes de pago, se actualiza su id, ya que su precio varia en función del plan contratado.

Una vez creado el modelo de datos, se ha añadido una relación entre el modelo recién creado *DomainSubscription* y un modelo ya existente llamado *Domain*, la cual ha sido de tipo *embedsOne*, es decir, que un Dominio solo puede tener una suscripción. Con la relación creada, el modelo *Domain* puede tener en cualquier momento un objeto de *DomainSubscription* con todos los campos recién definidos.

Para estar seguros de que los datos no son modificados vía web, se han tenido que desarrollar operativas de seguridad, en concreto, dos *hooks* (ganchos) del modelo *Domain* en el Backend, uno tipo *before access* (antes de acceder), para controlar que antes de acceder a los datos no se esté enviando al FrontEnd datos privados de la suscripción, y otro tipo *access* (acceso) para controlar que cuando se haga alguna petición a los datos de suscripción, no se modifique ningún campo del mismo, ya que eso solo debe ser permitido a través del código ya automatizado o vía base de datos manualmente por un administrador de Wattwin.

6.3.2 Identificación y creación del servicio

Como es necesario restringir el acceso a determinadas operativas internas de Wattwin, se ha necesitado desarrollar una directiva de Angular, con el nombre de *ngAccess*. Esta directiva comprueba que el *accessLevel* del Dominio sea más grande o igual que el *accessLevel* de la operativa, y en consecuencia restringe o da acceso, ya que en el caso de que el *accessLevel* del Dominio sea menor al de la operativa, esa operativa no será visible por el usuario. Todo este nuevo control es administrado por un servicio llamado *DomainSubscriptionService*, que ha sido creado exclusivamente para ello.

Además, es necesario gestionar el inicio de sesión a la web, ya que se quiere restringir el acceso a los usuarios sin suscripción, y personalizarles la página a los suscritos con prueba gratuita o con suscripción de pago. Para ello, se programó un control de suscripción en el login del BackEnd que comprueba si existe *trialExpirationDate* y *statusId*, en donde la primera ya ha sido explicada y la segunda corresponde al estado actual del Dominio. Existen cuatro posibles casos en función del valor de esos campos:

- **Sin trialExpirationDate y statusId ACTIVE**. Al no tener periodo prueba gratuita pero estado de suscripción activo, el usuario puede acceder a la web como suscriptor de pago.

- **trialExpirationDate vigente y statusId TRIAL**. Al existir una prueba de suscripción vigente, se entiende que el usuario está suscrito a un plan con prueba gratuita, por lo que tendrá acceso a la web con las mismas restricciones que ese plan.

- **trialExpirationDate caducado**. Al existir una fecha de periodo de prueba caducada, se entiende que el usuario no ha contratado ninguna suscripción de pago, por lo que se le denegará el acceso a la web.

- **statusId inactiva**. Que el estado de la suscripción sea inactivo quiere decir que no es ni ACTIVE ni TRIAL, pudiendo tener hasta 5 estados diferentes que representan fases de un Dominio, como son BOOTING (creando Dominio), HOLD (Dominio en espera), CANCELLED (suscripción cancelada), DELETED (suscripción eliminada) y PAUSED (suscripción pausada).

  En estos 5 estados de inactivo, se deniega el acceso al usuario, ya que, como se ha explicado antes, se quiere restringir el acceso a todo usuario no suscrito, aunque ya tenga el Dominio creado.

En los casos en donde se deniega el acceso a la plataforma y no existen más Dominios activos vinculados con el usuario, si la prueba gratuita se ha caducado se mostrará un mensaje de error en el login diciendo “Se ha caducado la prueba gratuita. Contacte con el administrador para contratar una.” y cuando no exista una suscripción activa se mostrará “No existe suscripción activa. Contacte con el administrador para contratar una.”.

Para tratar el caso de tener una periodo de prueba vigente, se ha tenido que crear el permiso *subscription_rw*, el cual solo es asignado al administrador de cada Dominio o espacio de trabajo, permitiéndole acceder a la interfaz de gestión de suscripción, la cual está únicamente disponible para los administradores de WorkSpace y será explicada más adelante en el documento. También, se le mostrará al administrador del WorkSpace un *banner* en la parte superior de la página con los días restantes para que caduque la prueba de suscripción.

En el caso de tener una suscripción de pago activa, el acceso será total a la web, únicamente restringido por el *accessLevel* del plan contratado. En este caso, también se le asignará el permiso *subscription_rw* al administrador del WorkSpace dándole acceso a la interfaz de suscripción, mostrando un *banner* como el del periodo de prueba si el plan contratado es el *Free*, informando del acceso limitado que comporta esa suscripción.

6.3.3 Diseño de interfaz y componentes

Llegado el momento de implementar las interfaces de suscripción, se decidió crear un componente con el nombre de *merchantSubscription* que contuviese una *card* o “tarjeta” desde la cual poder visualizar los datos de la suscripción, y además poder gestionar la propia suscripción clicando en un botón, desde donde se pueden realizar acciones tales como suscribirse, cambiar de plan, cancelar el plan o editar los datos de facturación. Este componente es llamado desde el

detalle de un Dominio (WorkSpace) accediendo a través del Dominio de Wattwin y en la configuración del propio espacio de trabajo, y solo será mostrado si se posee el permiso anteriormente mencionado, *subscription_rw*, que al afectar a módulos diferentes tubo que ser definido dos veces, una vez para cada uno de ellos.

El contenido de esta *card* es únicamente visible por el administrador del WorkSpace, a través de la cual tiene acceso a los datos de suscripción y al botón de gestión de la suscripción. En el caso del administrador de Wattwin, desde el propio Dominio de Wattwin puede acceder al listado de todos los espacios de trabajo y visualizar desde fuera el tipo de suscripción contratado por cada uno de ellos y la fecha de expiración de la prueba gratuita, si posee una. Si entra dentro del detalle de uno, se puede ver también la *card* con los datos de la suscripción y el botón para gestionar esa suscripción o incluso para suscribirse, en el caso de un Dominio inactivo.

6.3.4 Testing

Para realizar el testeo de esta tarea, se han realizado las siguientes pruebas:

- Modificar los datos de suscripción de la base de datos por algunos no válidos, para ver si así se rompe la integridad.
- Intentar modificar datos desde el código interno, sin tener permisos, pasando por los *hooks* (ganchos) creados.
- Probar la directiva *ngAccess* con valores fuera de los límites.
- Intentar hacer login a la página con diferentes combinaciones de *statusId* y *trialExpirationDate*.
- Cambiar de espacio de trabajo (WorkSpace) en donde uno de ellos tenga *trialExpirationDate* activo (debe aparecer el banner) y otro sin él.
- Cambiar datos dinámicamente y ver si se modifican en las *cards* de Dominio.

Los testeos no han generado grandes problemas, por lo que han sido fáciles de resolver.

6.4 Integración Chargebee

En este apartado se explica el trabajo realizado durante el desarrollo de cada una de las subtareas de la integración del software de facturación Chargebee, las cuales son *Estudio de su funcionamiento*, *Creación de endpoints*, *Creación registro de actividad* y *Testing*.

6.4.1 Estudio de su funcionamiento

Para la integración de la plataforma con el software de facturación Chargebee, primero de todo se ha tenido que estudiar la API para entender el funcionamiento y el tipo de llamadas que se deben hacer para vincular los datos necesarios. Después, se ha probado a crear datos de prueba y ver la relación entre ellos y como podían ser solicitados, ya que se crearon códigos de prueba totalmente en local para realizar llamadas asíncronas a Chargebee y ver lo que devolvía. También se ha estudiado la mecánica que tiene para personalizar facturas, los *checkouts* y *Portal Sessions* que devuelve (los cuales son explicados más adelante), así como los *custom fields* (campos personalizados), con los cuales se pueden añadir campos propios a los objetos que proporciona, como son los clientes (*Customers*), las suscripciones (*Subscriptions*), los planes (*Plans*) y las facturas (*Invoices*), entre otros.

6.4.2 Creación de endpoints

Las primeras implementaciones prácticas que se han hecho son desde el BackEnd han sido los *endpoints* (URLs del BackEnd que responden a peticiones) *createMerchantSubscription* y *getMerchantPortalSession*. El primero hace una llamada a Chargebee para solicitar un *checkout* de suscripción, que proporciona una interfaz propia de Chargebee para pedir los datos de facturación y crear una suscripción. Esta llamada requiere de pasarle la id del plan con el que se quiere vincular la suscripción, la cual viene dada del *pop up* (ventana) de suscripción en donde se muestran los planes disponibles. El segundo hace otra llamada a Chargebee, pero en este caso para solicitar el portal de facturación (*Portal Session*), en donde, de la misma manera que en el *checkout*, lleva a una interfaz propia de Chargebee, pero en este caso para poder editar la suscripción, pudiendo cambiar de plan, cancelar la suscripción o editar los datos de facturación.

También, se ha creado el *endpoint getPlans* para solicitar los planes a Chargebee y poder mostrarlos en el *pop up* de suscripción, de una manera más dinámica, al clicar sobre el botón de suscribirse. También se ha creado el *endpoint getSubscriptionData* para poder mostrar los datos de una suscripción que contiene Chargebee en la *card* mencionada anteriormente, y no solo los que se tenían en base de datos.

Además, se han necesitado desarrollar tres nuevos *endpoints* relacionados con los anteriores, pero para casos internos de la plataforma, como es el control de eventos que se explicará más adelante. Al primero se le ha llamado *getItemById*, y sirve para solicitar un plan (*item*) en función de su id, muy útil para cuando solo se quieren recibir los datos de un plan y no de todos los planes, como hace el *endpoint getPlans*. Al segundo se le ha llamado *getItemPriceById*, sirve para solicitar los datos del precio de un plan (*item price*) en función de su id, cuya id es la utilizada para suscribir un Dominio. Al tercero se le ha llamado *createSubscriptionForItems*, su función es similar a la del *createMerchantSubscription*, pero sin necesidad de devolver un *checkout* de suscripción, ya que se le pasan los datos del *item* y la id del *customer* y crea la suscripción instantáneamente.

Por último, se ha creado un *endpoint* con el nombre de *merchantWorker* para registrar la actividad del *webhook* (gancho web) de Chargebee, el cual envía datos en formato JSON cada vez que se realiza una acción seleccionada. Su función es recibir los eventos generados por el *webhook* configurado en Chargebee con el nombre de *Merchant Webhook*, para poder registrar dentro de la base de datos todos los cambios en las suscripciones y actualizar la vista.

6.4.3 Creación registro de actividad

Para poder tener el registro de actividad (mencionado en el subapartado anterior) visible en la página, se ha creado una nueva *tab* (pestaña) llamada *Activity* en *Mi WorkSpace Módulos* en todos los espacios de trabajo, registrando todas las acciones realizadas por cada Dominio. Esta implementación funciona gracias al registro que guarda un nuevo *endpoint* que es explicado más adelante, con el nombre de *merchantNotification*.

Por otro lado, se ha creado una nueva gráfica en la sección de Dashboard (panel) en el listado de Dominios (únicamente visible desde el Dominio Wattwin), la cual es mostrada al filtrar por tipo de suscripción, y se visualiza como un diagrama de sectores con los diferentes tipos de suscripción (*subscriptionType*) de entre todos los Dominios de la plataforma.

6.4.4 Testing

Para realizar el testeo de esta tarea, se han realizado las siguientes pruebas:

- Crear en Chargebee diferentes planes y configuraciones para probar como actúa el sistema frente a ellos.
- Hacer llamadas a Chargebee con datos inconsistentes e ids inventadas.
- Probar el envio de eventos con menos datos de los necesarios al *endpoint merchantWorker* para ver su control de errores.
- Realizar acciones variadas en diferentes Dominios para comprobar que el registro de actividad se guarda donde debería.

Durante los testeos, han surgido diferentes problemas con la gestión de errores en el *merchantWorker*, los cuales, después de añadir un control de errores más preciso, se han conseguido solucionar.

6.5 Gestión de consumo

En este apartado se explica el trabajo realizado durante el desarrollo de cada una de las subtareas que han conllevado la gestión de consumo, las cuales son *Control de facturación*, *Definición tipos de consumo*, *Implementación control de consumo* y *Testing*.

6.5.1 Control de facturación

Para poder gestionar el consumo de los clientes y facturar en relación a ello, se ha tenido que añadir el control de eventos de Chargebee relacionados con la facturación y los *addOns* (extensiones de pago) a cobrar al final del término.

Al haber tenido que añadir el control de más eventos, con la duda de sí la asincronicidad de las llamadas de Chargebee podría provocar un desorden en las llegadas de los eventos, se ha decidido desarrollar una cola FIFO (First In First Out - El primero que entra es el primero que sale) que reciba los eventos de Chargebee y los meta en la cola, además de registrarlos en la *tab Activity* mencionada anteriormente.

Para que esta nueva cola pueda entrar en uso, se ha tenido que desarrollar un nuevo *endpoint* con el nombre de *merchantNotification*. Este *endpoint* logra capturar los eventos de Chargebee al igual que su versión anterior, pero en vez de procesarlos, los mete en la cola y devuelve los datos necesarios para que se registren correctamente en la *tab Activity*. Esta forma de tratar los eventos tiene una particularidad adicional, y es que permite poder registrar las acciones de Chargebee con el código que las procesa en local, es decir, únicamente requiere de tener el código del *merchantNotification* subido a *int* (entorno de testeo de primera fase), para que el *endpoint* de Chargebee pueda enviar ahí siempre los eventos y extraer desde el código local el contenido de la cola.

Por otra parte, para permitir al cliente la visualización de su consumo, el portal de facturación de Chargebee ofrece la opción de visualizar las facturas antiguas, así como las de los próximos meses, por lo que el cliente tiene total libertad para visualizarlas y, si quiere, descargarlas.

6.5.2 Definición tipos de consumo

Para la gestión del consumo del cliente y sus costes variables, se han definido los tipos de consumo (*addOns*) que se tendrán en cuenta a la hora de facturar al cliente.

- **Por usuarios creados**. Por cada usuario vinculado con el Dominio (*WorkSpace*) suscrito, se sumarán para cada ciclo de facturación y se añadirán a la factura como una carga adicional.
- **Por Powerplant creadas**. Por cada kilovatio (Kw) consumido en las plantas fotovoltaicas (*Powerplants*) creadas por el usuario/s dentro de su Dominio, se le añadirá como coste variable a la factura, con la particularidad de que se le cobrarán siempre las creadas durante ese ciclo de facturación, no todas las creadas en total (como es en el caso de los usuarios).

Estos nuevos *addOns* creados se han vinculado con los metadatos de cada uno de los planes, asignando a cada uno un precio personalizado.

6.5.3 Implementación control de consumo

Primero de todo, se ha actualizado el *endpoint merchantWorker*, mejorado la gestión de la creación y cambio de una suscripción, para así añadir en base de datos la información del plan junto con la id de los *addOns* a consumir, así como información de que cantidad ya ha sido cobrada y la fecha de la misma. Esto ha conllevado la creación de un nuevo modelo de datos de tipo *transient* llamado *DomainSubscriptionAddon*, al cual se le ha asignado una relación *hasMany* (una suscripción puede contener múltiples *addOns*) con el modelo anteriormente creado, *DomainSubscription*.

También, se le han añadido al *Merchant Webhook* los eventos relacionados con los pagos, así como su gestión en el *endpoint merchantWorker*, tanto para los pagos fallados como con éxito. También se ha añadido el evento de generación de nueva factura, la cual se ha configurado en Chargebee para quedarse en estado pendiente y así poder añadirle los costes variables de los *addOns*. Por lo tanto, una vez

llegado el evento de factura pendiente, se le añaden los *addOns* consumidos a la factura del cliente y se actualiza esa información en base de datos.

6.5.4 Testing

Para realizar el testeo de esta tarea, se han realizado las siguientes pruebas:

- Generar diferentes facturas sobre un mismo Dominio y verificar que se generan correctamente.
- Acceder al portal de un Dominio y visualizar el historial de facturas, para verificar que coincidan con las generadas anteriormente.
- Hacer diferentes combinaciones de *addOns* en un plan y verificar que el consumo se calcula correctamente.

Una vez realizados los testeos, se han podido encontrar ciertos bugs con el cálculo de los *addOns* variables, los cuales han sido solucionados sin problemas.

6.6 Testing final

En esta tarea se exponen y explican las diferentes pruebas de testeo realizadas para garantizar el correcto funcionamiento del servicio programado. Esta conformada por las subtareas *Definición pruebas de testeo*, *Ejecución pruebas de testeo* y *Solución de bugs*.

6.6.1 Definición pruebas de testeo

Primero de todo, se han definido las pruebas de testeo para verificar que todo funciona según lo esperado. Las pruebas definidas han sido las siguientes:

- Crear un Dominio desde cero vinculado con una suscripción gratuita y asignándolo al usuario test, para así poder acceder y navegar libremente por el Dominio.
- Cambiar de plan en modo TRIAL solicitando una suscripción de pago, para verificar así la factura generada por el cambio.
- Realizar cambios entre suscripciones de pago desde el propio Dominio, asignando diferentes combinaciones de *addOns* y alterando el ciclo de facturación, para ver si las facturas generadas son correctas.
- Crear *Powerplants* y vincular diferentes usuarios con el Dominio.
- Generar varias facturas, alterando de nuevo el ciclo de facturación, para ver que los costes variables se cobran correctamente.
- Realizar pagos reales en el entorno de *live* (para usuarios finales), para verificar que la facturación se puede llevar a cabo correctamente.

6.6.2 Ejecución pruebas de testeo

Cada una de las pruebas de testeo ha sido ejecutada en repetidas ocasiones, lo que ha conllevado la creación de múltiples nuevos Dominios y muchas facturas de prueba. Este testeo se ha llevado a cabo un poco antes de lo previsto, ya que se preveía muy útil para encontrar *bugs* previos a la subida del código a entornos superiores.

6.6.3 Solución de bugs

Después de una búsqueda exhaustiva de *bugs*, se han logrado encontrar y resolver todos ellos gracias a las pruebas de testeo definidas en el apartado 6.6.1. Por eso mismo, esta tarea se puede considerar un éxito.

7 PRESENTACIÓN Y DISCUSIÓN DE RESULTADOS

El proyecto basado en el Módulo de Gestión del Consumo (MGC) de la plataforma Wattwin ha sido desarrollado con éxito, a la espera de subirlo definitivamente a producción y testearlo con usuarios finales. Se ha logrado automatizar el proceso de gestión de suscripciones y consumos, proporcionando a la plataforma un nuevo e importante avance en la evolución de su servicio web. El servicio desarrollado es ahora mismo capaz de permitir al cliente:

- Crear una suscripción gratuita vinculada con un Dominio (*WorkSpace*) para poder acceder a la plataforma y ver de primera mano los servicios que ofrece.
- Suscribirse de pago para poder optar a operativas y funcionalidades más avanzadas, teniendo la posibilidad de cambiar de plan de suscripción, además de añadir cargos o *addOns* a la misma.
- Gestionar y visualizar los datos de su suscripción, añadiendo cargos variables y pudiendo personalizarla.
- Visualizar sus facturas, así como los datos de consumo, los cuales pueden ser visualizados en la factura, que son visibles desde el portal que ofrece el software de facturación Chargebee.
- Visualizar el *banner* de suscripción en periodo de prueba cuando la suscripción esté en modo TRIAL y el plan gratuito cuando se tiene contratado el plan *Free*.

De los resultados expuestos, también están, aunque menos visibles, la automatización de la gestión interna de la empresa sobre los datos de los clientes. Todas estas gestiones han conllevado muchas horas de trabajo, las cuales, probablemente, se hayan visto aligeradas gracias al previo conocimiento de la mayoría de lenguajes y metodologías utilizadas, ya que distintas competencias adquiridas en el grado han ayudado a tener una visión más abierta del funcionamiento interno de un servicio web público, como es Wattwin.

Que los resultados hayan sido los esperados, no quiere decir que hayan salido de la forma en que se esperaba, ya que, de primeras, se pretendía desarrollar una interfaz propia de gestión del consumo y así poder visualizar el consumo de los clientes y sus facturas dentro de la plataforma Wattwin. Finalmente, el objetivo de tener un sistema de gestión de consumo ha sido cumplido, pero delegandole gran parte del trabajo a Chargebee, ya que la estimación de dificultad y complejidad de esta implementación no fue prevista en un inicio. También han surgido contratiempos como ha sido la migración de versión de la API de Chargebee, ya que, en mitad del desarrollo del proyecto, sacaron una nueva versión de su catálogo, provocando cambios en implementaciones ya terminadas, aunque finalmente no haya afectado a los periodos establecidos en las tareas.

Queda claro después de este largo periodo, que este proyecto era alcanzable para un estudiante de Ingeniería Informática sin experiencia previa en el mundo laboral. Eso sí, la altura e importancia del servicio desarrollado no hubiera sido posible realizarla solo, ya que los recursos proporcionados por la empresa y la ayuda recibida lo han hecho mucho más alcanzable.

8 Conclusiones

La empresa de servicios tecnológicos basada en energía fotovoltaica, SIIE, la cual no disponía de ningún control sobre el consumo de los clientes más que las propias facturas que se debían generar manualmente al final de cada periodo, ha conseguido, gracias a este proyecto, automatizar el proceso de facturación y gestión del consumo de sus clientes.

Para desarrollar esta nueva gestión, se han tenido que definir los diferentes requisitos y objetivos del proyecto, los cuales han sido traducidos en tareas que han comportado la definición de los planes y las jerarquías de suscripción, la integración del software de facturación Chargebee y la gestión del consumo vinculado, y por último, el testeo de todo el sistema en conjunto. Todas estas tareas se han podido lograr en el tiempo estimado de duración del TFG, con la particularidad de estar enfocada a usuarios reales, algo que hubiese sido muy diferente en un proyecto personal no vinculado a una empresa. Este proceso me ha ayudado a aprender del funcionamiento interno del desarrollo de un software como servicio (SaaS) y de la forma de trabajar de un equipo de desarrollo web desde dentro, así como todas las fases de implementación y test que se llevan a cabo antes de subir un código a producción.

Esta nueva gestión implementada, ha permitido tener el control sobre las acciones de todos los usuarios y poder guardar un registro internamente. Todo este nuevo flujo facilita la gestión de los datos y ayuda a delegar trabajo a otros softwares que automatizan aún más los procesos internos, como ha sido el caso del software de facturación Chargebee. En breve todo el código desarrollado será subido a producción y se podrá empezar a utilizar libremente con usuarios finales.

Independientemente del buen resultado final, han habido decisiones y objetivos iniciales que finalmente se decidieron apartar del proyecto, tales como la gestión de las suscripciones y planes sin depender de un software externo, la creación de una vista propia para que cada usuario administrador pueda consultar sus datos de consumo y gráficas asociadas, y también un sistema de filtrado y alarmas por fechas para las facturas y consumos. Aunque Chargebee proporcione algunas funcionalidades parecidas, la idea inicial fue desarrollarlas en Wattwin directamente, pero posteriormente se cambio el enfoque del proyecto hacía Chargebee, debido a la alta complejidad que conllevaba el desarrollo de un diseño propio.

Aún habiendo cumplido con todos los objetivos propuestos una vez acabada su definición, existen avances y mejoras del servicio que se irán cubriendo más adelante. En los siguientes meses se pretende profundizar más en la restricción de acciones por plan, dándole más importancia, ya que es una funcionalidad muy importante para el negocio y la venta de planes personalizados. También se pretenden abordar en un futuro los objetivos que no entraron en el proyecto y que fueron apartados en un inicio (mencionados en párrafo anterior), y sin olvidarnos de en un futuro no muy lejano sustituir el software de facturación Chargebee por un servicio propio de Wattwin que gestione todo el consumo sin depender de un software externo, logrando una mejora cuantitativa en el desarrollo de la plataforma.

Agradecimientos

Quiero agradecer primero de todo a mi familia, ya que me ha apoyado en todo momento en esta nueva etapa de mi vida, y a mis amigos por aguantarme en los malos momentos. También me gustaría agradecer a la empresa SIIE y a su equipo de trabajo todo este valioso periodo que he compartido con ellos y que me ha ayudado a crecer como programador web, dándome todas las facilidades que necesitase para poder terminar este proyecto en el tiempo estimado. Por último, pero no menos importante, agradezco a mi tutor del trabajo, Ramón Martí, por prestarme atención cuando tenía dudas y por guiarme hacia la creación de un documento profesional adecuado para un TFG.

Referencias

[1] SIIE, Consultancy and BPO services for the solar PV industry. [En línea]. Disponible: http://sii-e.com/

[2] Wattwin, La solución digital para el diseño, venta y gestión de instalaciones fotovoltaicas. [En línea]. Disponible: https://wattwin.com/

[3] Chargebee, The subscription management platform that enabled Slidebean to slide into 30+ countries. [En línea]. Disponible: https://www.chargebee.com/docs/

[4] CBA, Component-Based Architecture. [En línea]. Disponible: https://medium.com/@dan.shapiro1210/understanding-component-based-architecture-3ff48ec0c238

[5] TypeScript, Typed JavaScript at Any Scale. [En línea]. Disponible: https://www.typescriptlang.org/

[6] Angular, Manual de Angular. [En línea]. Disponible: https://desarrolloweb.com/manuales/manual-angular-2.html

[7] Pug, A Beginner's Guide to Pug. [En línea]. Disponible: https://www.sitepoint.com/a-beginners-guide-to-pug/

[8] Sass, Sass Tutorial. [En línea]. Disponible: https://www.w3schools.com/sass/default.php

[9] Promises, Async Await in Node.js. [En línea]. Disponible: https://blog.risingstack.com/mastering-async-await-in-nodejs/

[10] LoopBack, LoopBack 3.x Documentation. [En línea]. Disponible: https://loopback.io/doc/en/lb3/

[11] MongoDB, MongoDB CRUD Operations. [En línea]. Disponible: https://docs.mongodb.com/manual/crud/

[12] Elastic, Kibana: tu ventana al Elastic Stack. [En línea]. Disponible: https://www.elastic.co/es/kibana

APÉNDICE

A.1 Diagramas de Casos de uso

En está sección del apéndice se muestran los diferentes diagramas de casos de uso del proyecto, diferenciando entre los de un administrador de Wattwin y de WorkSpace.

A.1.1 Consumo para administrador de WorkSpace

En el diagrama Fig.4 se muestran las acciones relacionadas con el consumo que puede realizar el administrador de cada WorkSpace, es decir, cada usuario en su espacio de trabajo. Las acciones o casos de uso a realizar incluyen visualizar el consumo actual y el historial del mismo.

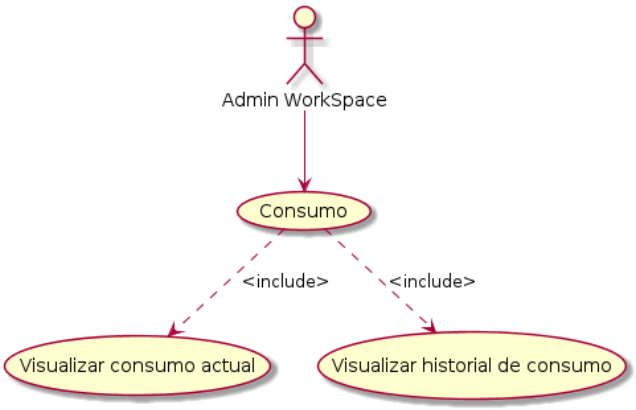

Fig. 4: *Diagrama de casos de uso de Consumo para un administrador de WorkSpace.*

A.1.2 Plan para administrador de WorkSpace

En el diagrama Fig.5 se muestran las acciones relacionadas con la gestión del plan contratado, el cual debe poder ser mejorado, degradado o dado de baja, así como poder visualizar los precios de cada uno. Además de poder añadir *addOns* o costes variables a su suscripción de un plan. Diagrama que va directamente relacionado con el objetivo de *Integración de Chargebee*.

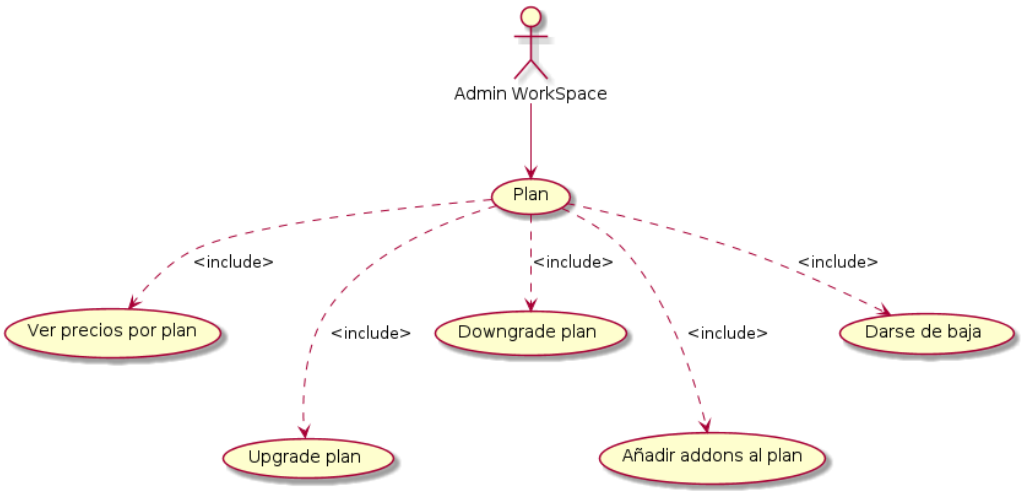

Fig. 5: *Diagrama de casos de uso de Plan para un administrador de WorkSpace.*

A.1.3 Facturación y Cobro para administrador de WorkSpace

En el diagrama Fig.6 se muestran las acciones relacionadas con la facturación y el cobro de la suscripción y sus respectivos costes variables. Algunas de las acciones o casos de uso que conforman este diagrama son la visualización de la factura actual y el historial de facturas, la opción de descargarse las facturas o editar los datos de facturación, entre otros. Diagrama que va directamente relacionado con el objetivo de *Integración de Chargebee*.

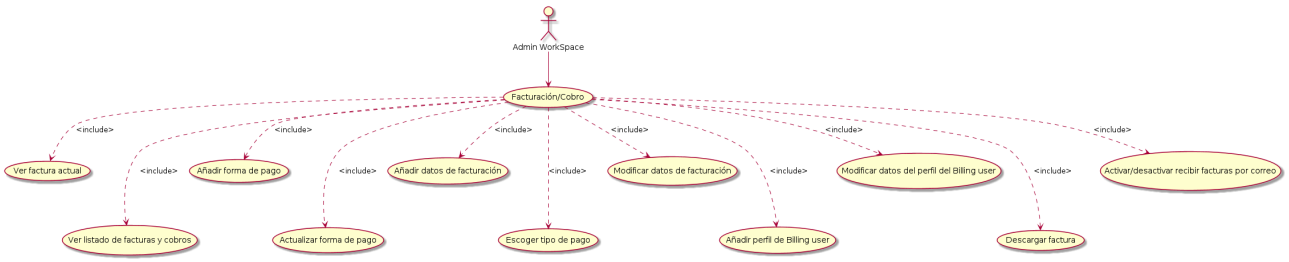

Fig. 6: *Diagrama de casos de uso de Facturación y Cobro para un administrador de WorkSpace.*

A.1.4 Consumo para administrador de Wattwin

En el diagrama Fig.7 se muestran las acciones de un administrador de Wattwin, es decir, un administrador interno de la empresa SIIE, el cual hereda todas las acciones de un administrador de WorkSpace y le suma, en este caso, acciones de consumo como son añadir, eliminar y modificar un tipo de consumo, y realizar seguimientos de consumo de los WorkSpaces.

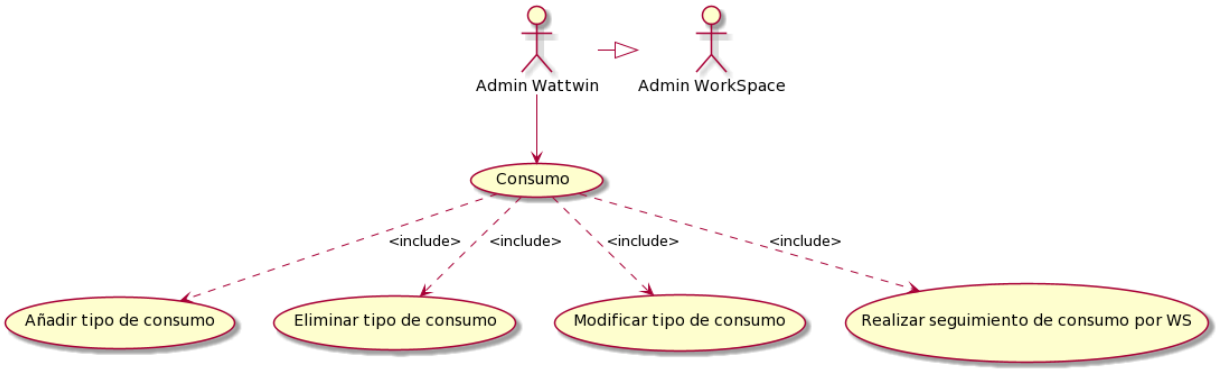

Fig. 7: *Diagrama de casos de uso de Consumo para un administrador de Wattwin.*

A.1.5 Plan para administrador de Wattwin

En el diagrama Fig.8 se muestran más acciones del administrador de Wattwin, heredando las del administrador de WorkSpace, las cuales ahora están enfocados en la gestión de planes, como son la creación, modificación y eliminación de un plan, la creación y eliminación de una operativa, y la limitación de operativas o acciones por plan.

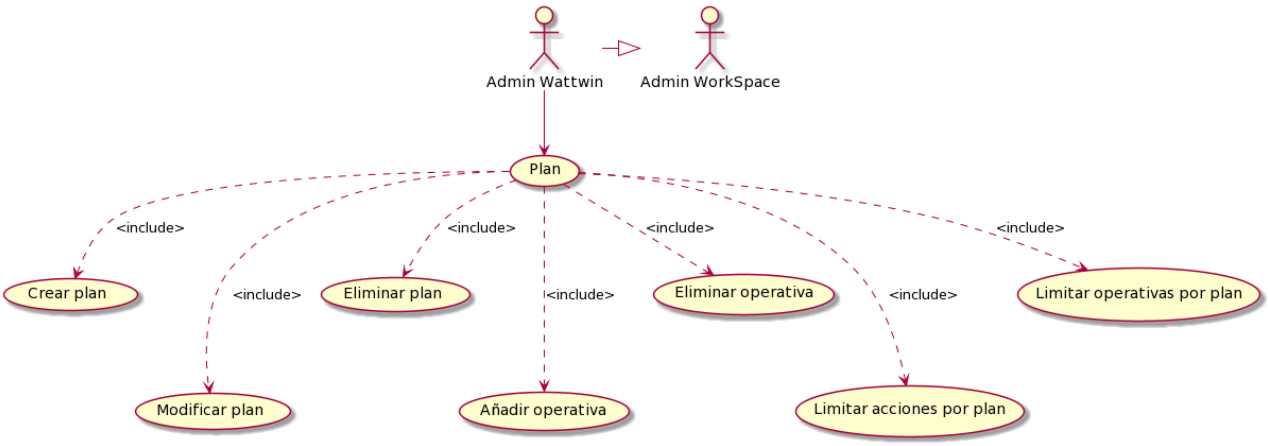

Fig. 8: *Diagrama de casos de uso de Plan para un administrador de Wattwin.*

A.1.6 Tarifas para administrador de Wattwin

En el diagrama Fig.9 se muestran más acciones del administrador de Wattwin, heredando las del administrador del WorkSpace, las cuales ahora están enfocadas en la gestión de las tarifas. Los casos de uso consisten en crear, modificar y eliminar una tarifa, así como gestionar la asociación de tarifas a los planes.

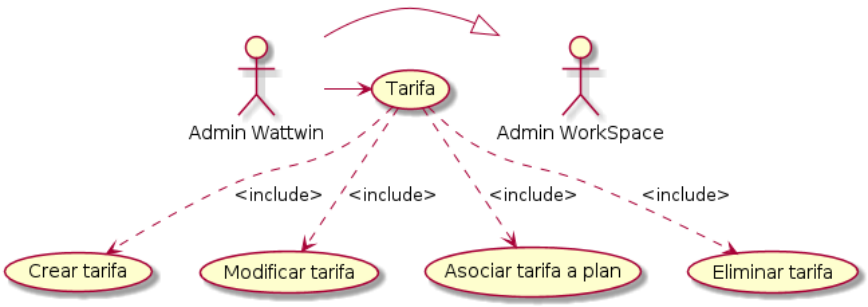

Fig. 9: *Diagrama de casos de uso de Tarifas para un administrador de Wattwin.*

A.1.7 Facturación y Cobro para administrador de Wattwin

En el diagrama Fig.10 se muestran más acciones del administrador de Wattwin, heredando las del administrador del WorkSpace, las cuales ahora tratan sobre la facturación y el cobro. Los casos de uso consisten en gestionar las facturas, así como añadirles cargos y cerrarlas.

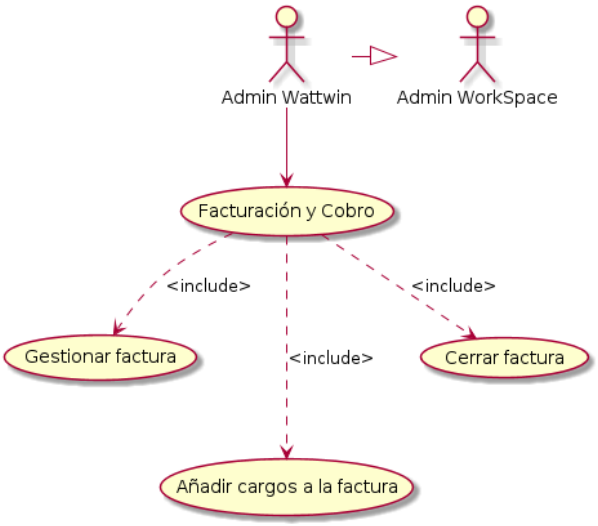

Fig. 10: *Diagrama de casos de uso de Facturación y Cobro para un administrador de Wattwin.*

A.2 Diagrama de Gantt

En está sección del apéndice se muestra el diagrama de Gantt de este proyecto en su fase más temprana A.2.1 y en su fase final A.2.2.

A.2.1 Diagrama de Gantt inicial

En Fig.11 se puede ver el diagrama de Gantt inicial del proyecto.

A.2.2 Diagrama de Gantt modificado

En Fig.12 se puede ver el estado del diagrama de Gantt después de los cambios sufridos a lo largo del proyecto.

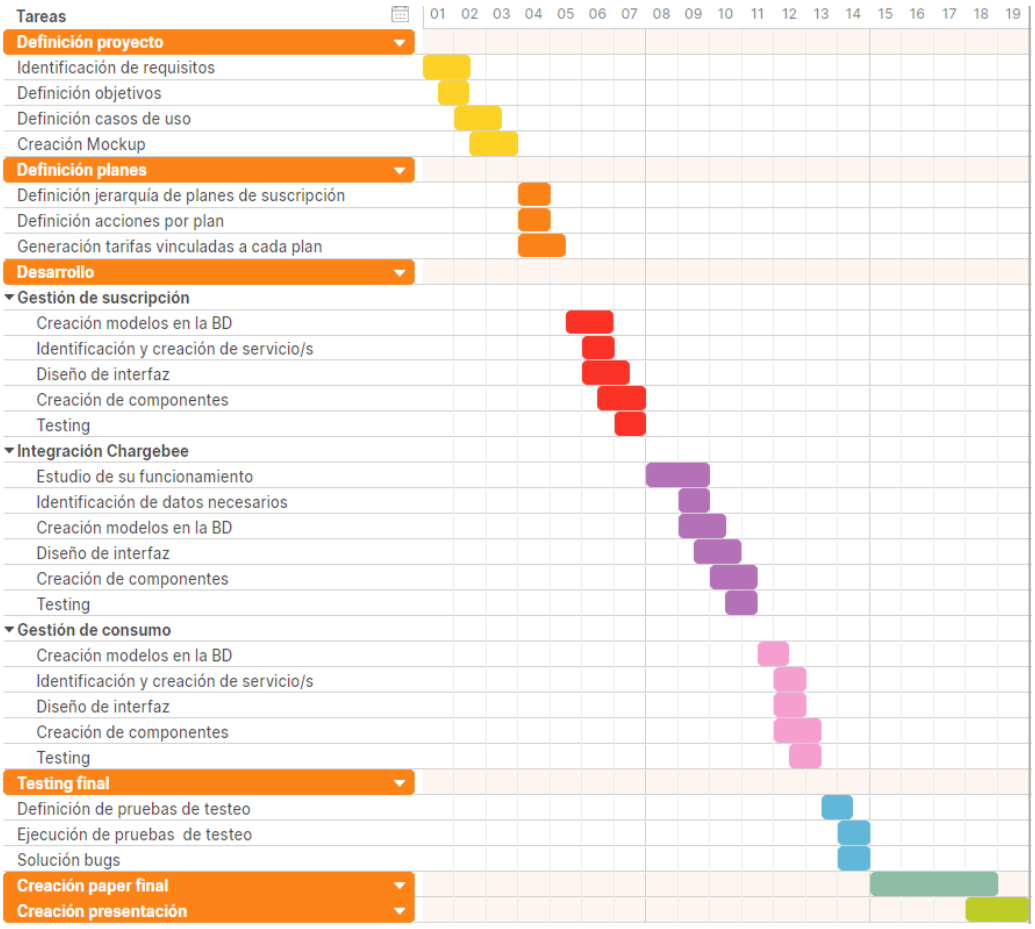

Fig. 11: *Diagrama de Gantt inicial.*

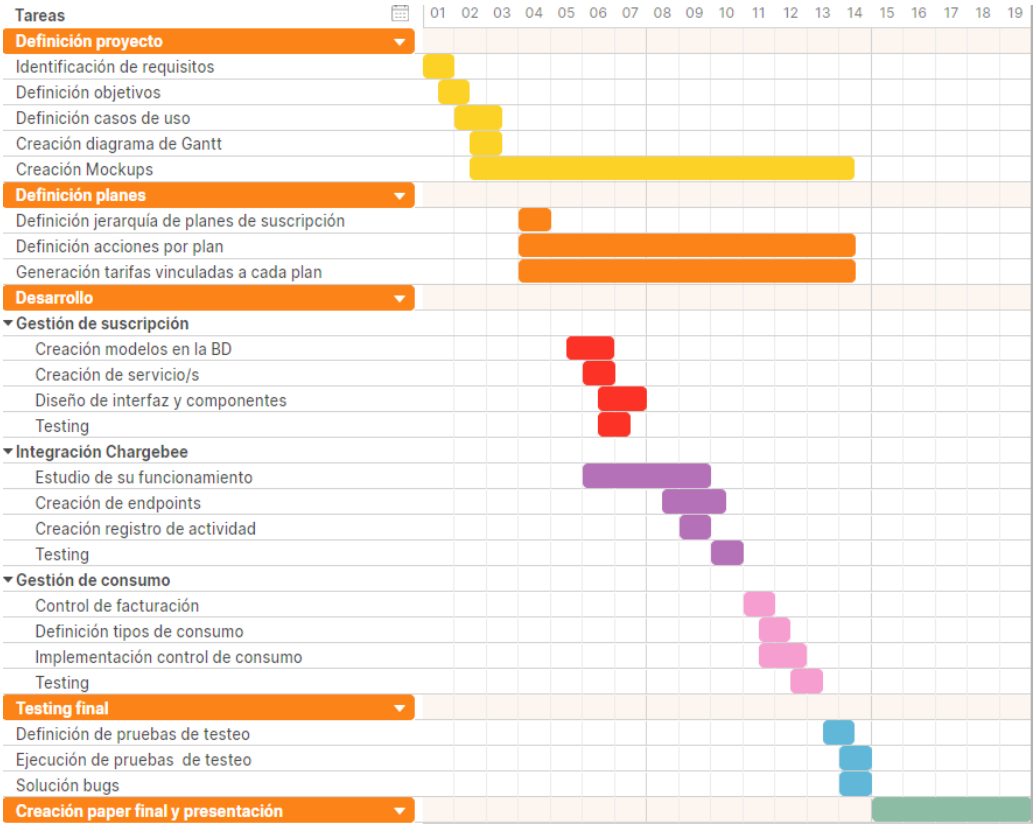

Fig. 12: *Diagrama de Gantt modificado.*